A documentation of the

Face Detection and Quick Response Code Scanner System

A partial fulfillment of the requirements for the midterm of first semester in the course Modelling and Simulation

Polytechnic University of the Philippines

Department of Computer and Information Sciences

BSCS 2-3

Group 4

Table of Contents

Members and Developers

Project Manager and System Developer

Calendario, Mark Kenneth

User Interface and User Experience Designers

Abarre, Jeo

Sinalubong, Sean Matthew

System Analyst

Sinalubong, Sean Matthew

Business Analyst

Cruz, Jan Miles

Technical Designers

Gani, Zaimmon

Yim, Gwyneth Ann Marie

Technical Writers

Brigola, Jaycee

Vedasto, Micole Aaron

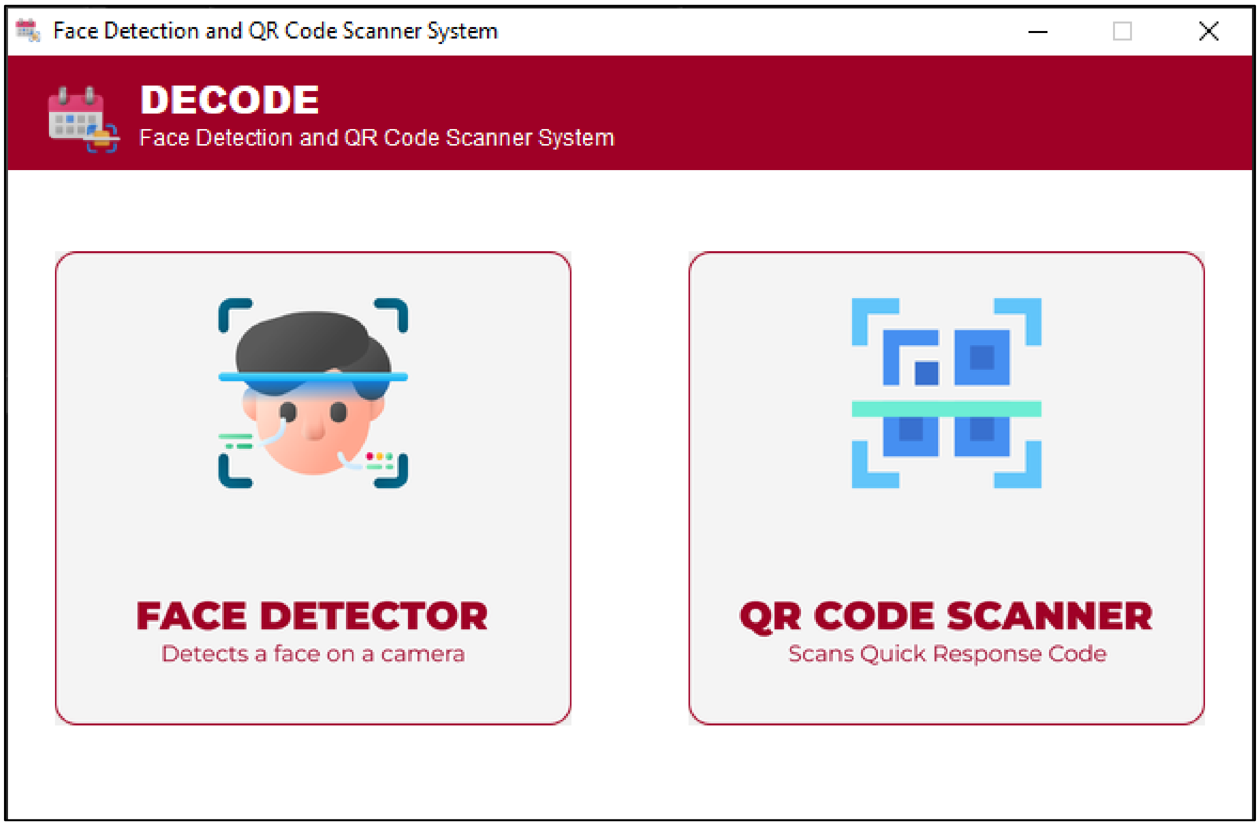
Project Description

The QR Code Scanner and Face Detection System is a combination of two useful features: quick response code scanning and decoding and frontal face detection. The QR code scanning feature allows users to decode and get information encoded in a machine-readable black and white matrix by simply pointing their device's camera at a QR code, while the face detection feature uses advanced computer vision algorithms to detect and track faces in real-time. This project is designed to be user-friendly, with a simple and intuitive interface.

System Snapshot

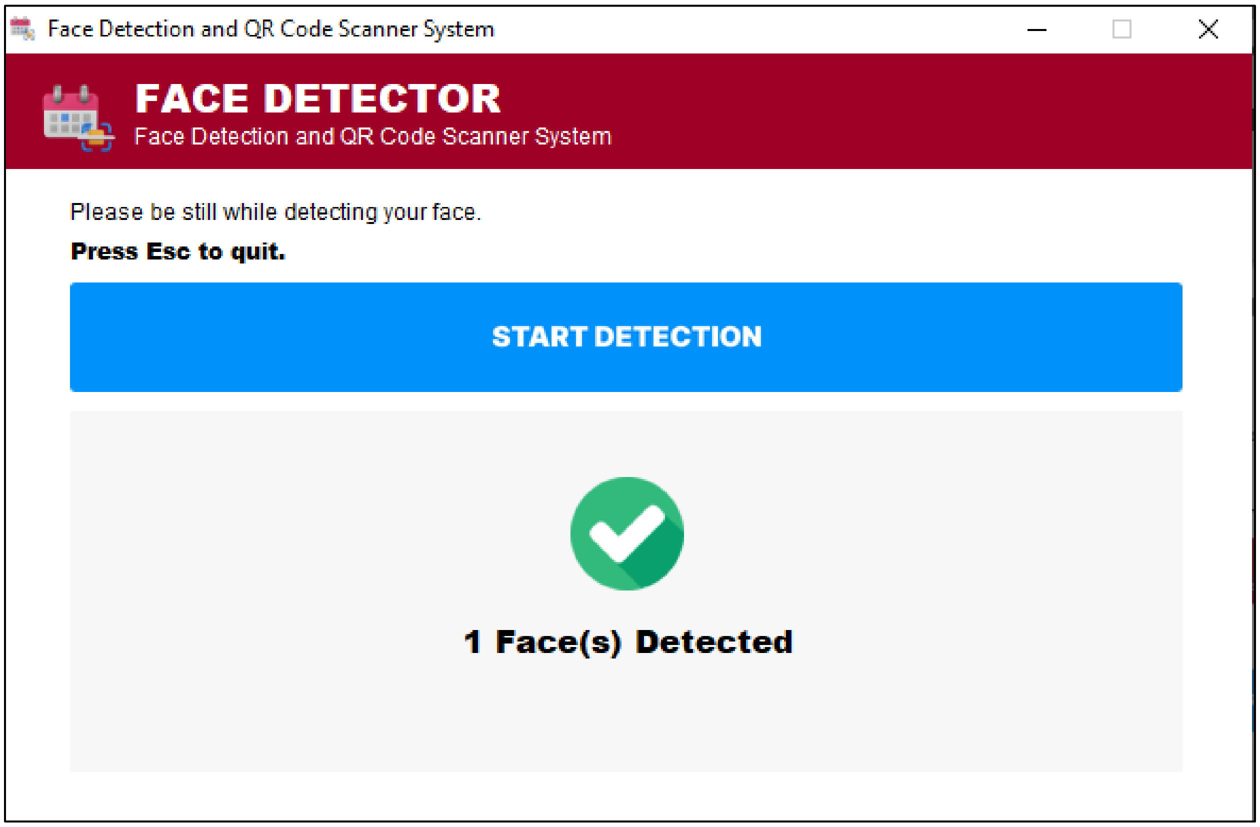
Main Frame of the System

Upon launching the application by clicking on the executable file (exe), the main frame of the application will appear. The root layer of the graphical user interface contains two buttons for the main features of the system: a face detector and a QR code scanner.


Face Detection and QR Code Scanner System
DECODE
Face Detection and QR Code Scanner System
FACE DETECTOR
Detects a face on a camera
QR CODE SCANNER
Scans Quick Response Code

Face Detection Data Frame

This Face Detector frame is stackable from the root layer of the GUI. It contains the control for starting the detection and the guides on how to use and exit the frame. This frame is responsible for displaying the real-time state of the camera device, face detection, and faces count on the frame. The start detection button is placed on this frame for starting up a real-time camera stream wherein the face detection will take place.


Face Detection and QR Code Scanner System

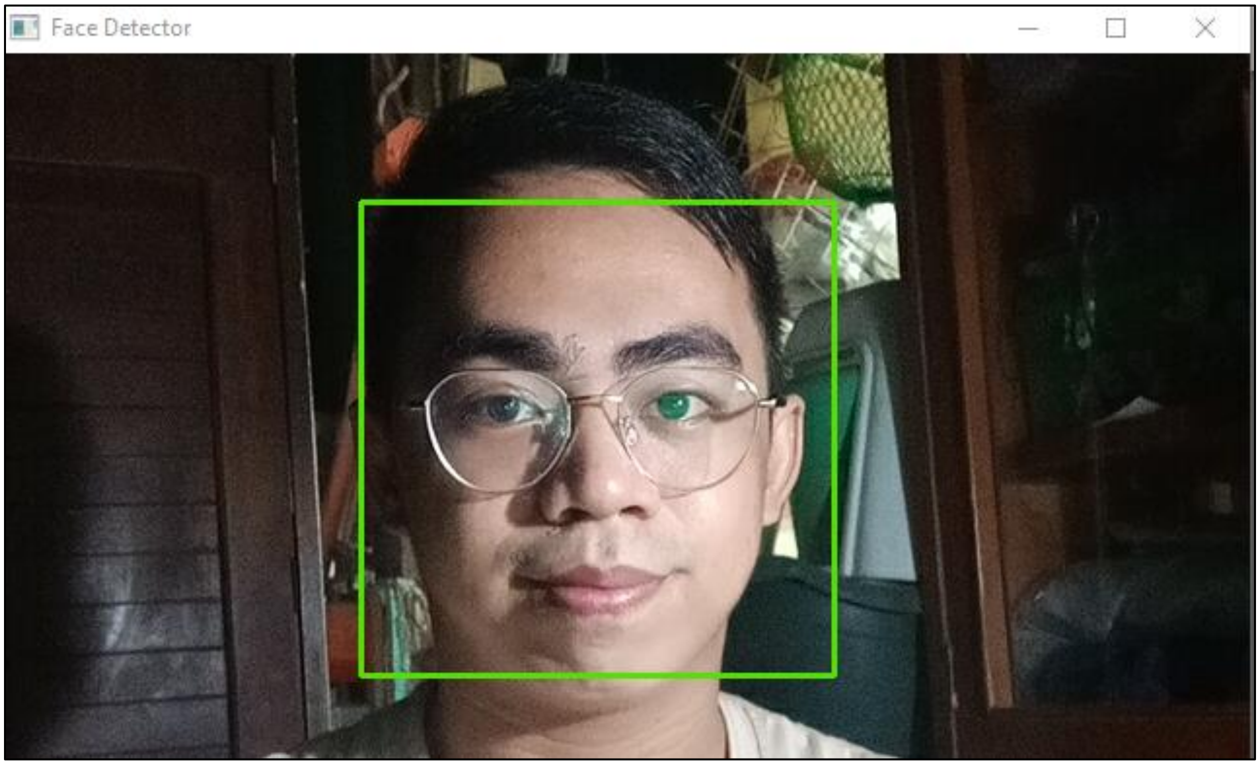
FACE DETECTOR

Face Detection and QR Code Scanner System

Please be still while detecting your face.

Press Esc to quit.

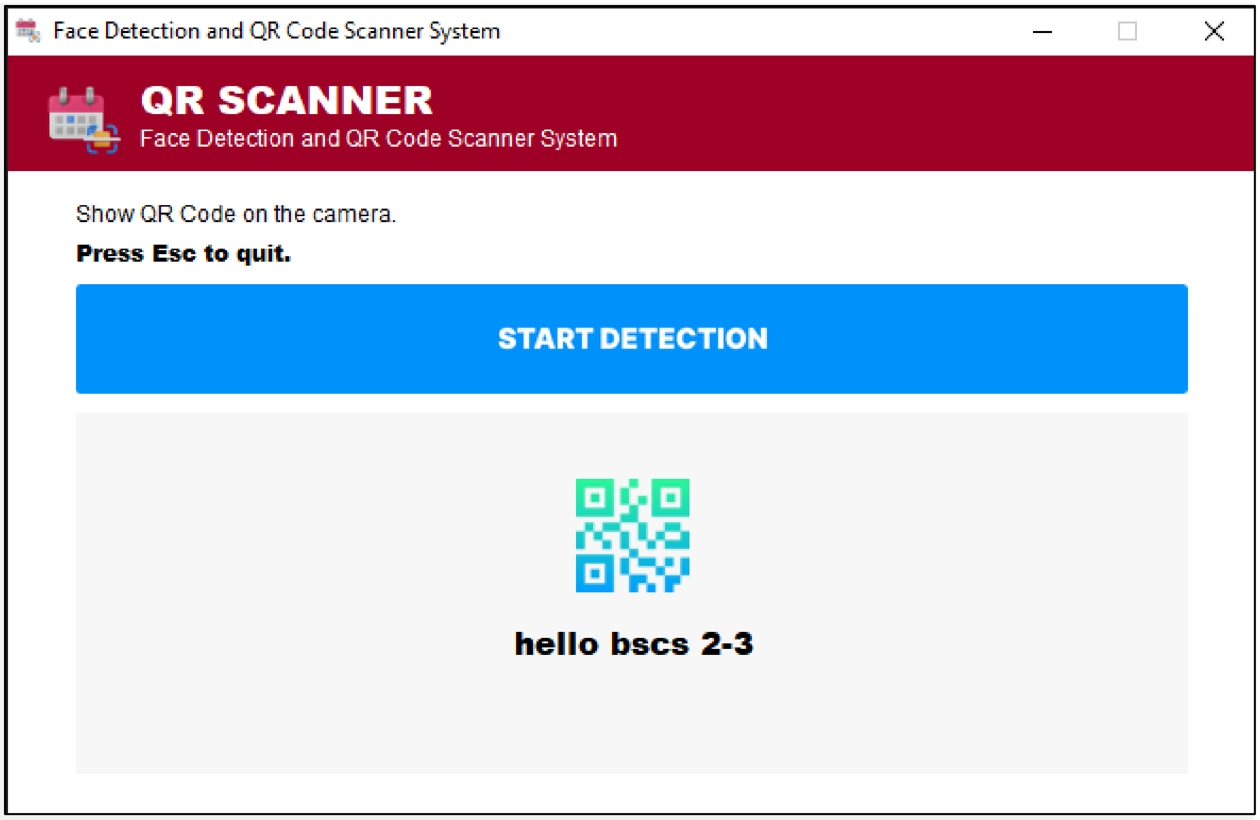
1 Face(s) Detected



QR Code Scanner Data Frame

This QR Code Scanner Data Frame is also stackable from the root layer. Once the QR Code Scanner button was clicked from the main frame, the frame below will popup. This frame consists of real-time data of decoded QR code and other indicators from the camera stream.

The start detection button is placed and packed on this frame for showing up a real-time camera window for the QR Code scanning to take place.


Face Detection and QR Code Scanner System

QR SCANNER

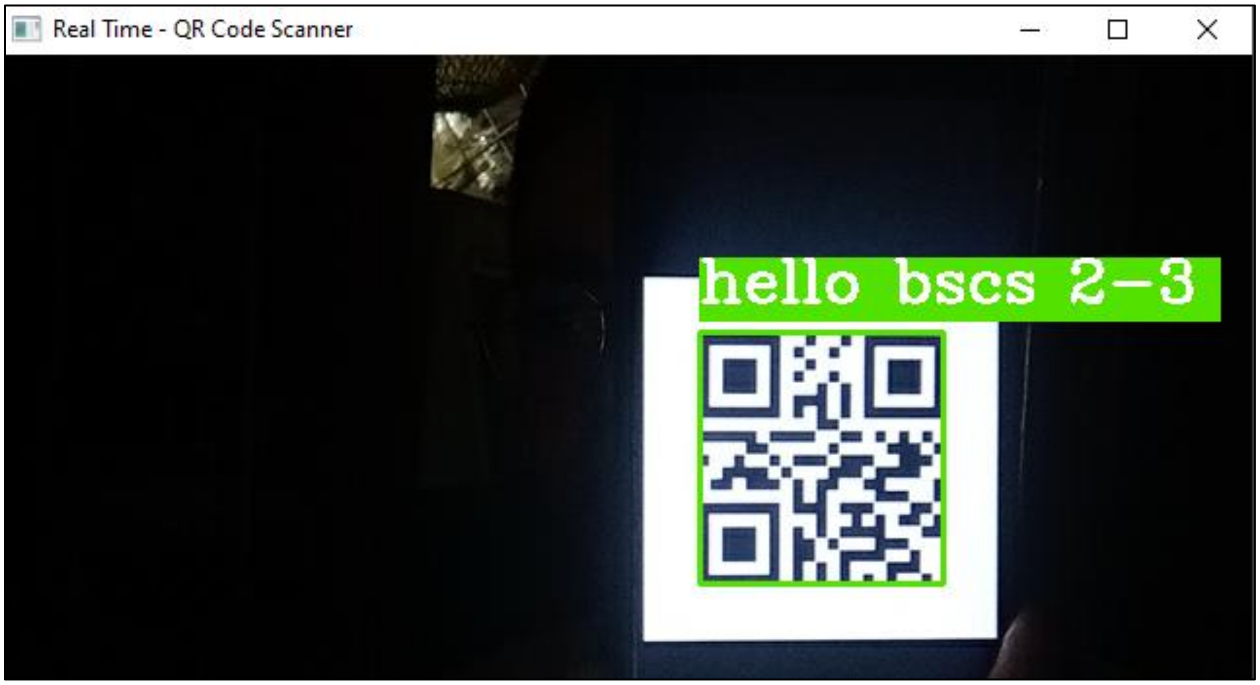
Face Detection and QR Code Scanner System

Show QR Code on the camera.

Press Esc to quit.

START DETECTION

hello bscs 2-3



System Pseudocode and Flowchart

This portion of this paper will discuss and show the flowchart or diagram representation of the system and pseudocode or flow of the system Face Detection and Quick Response Code Scanner System in text format. The flowchart and pseudocode of the system is divided into three parts:

1. Main Panel of the System
2. Face Detection Data Frame
3. QR Code Scanner Data Frame

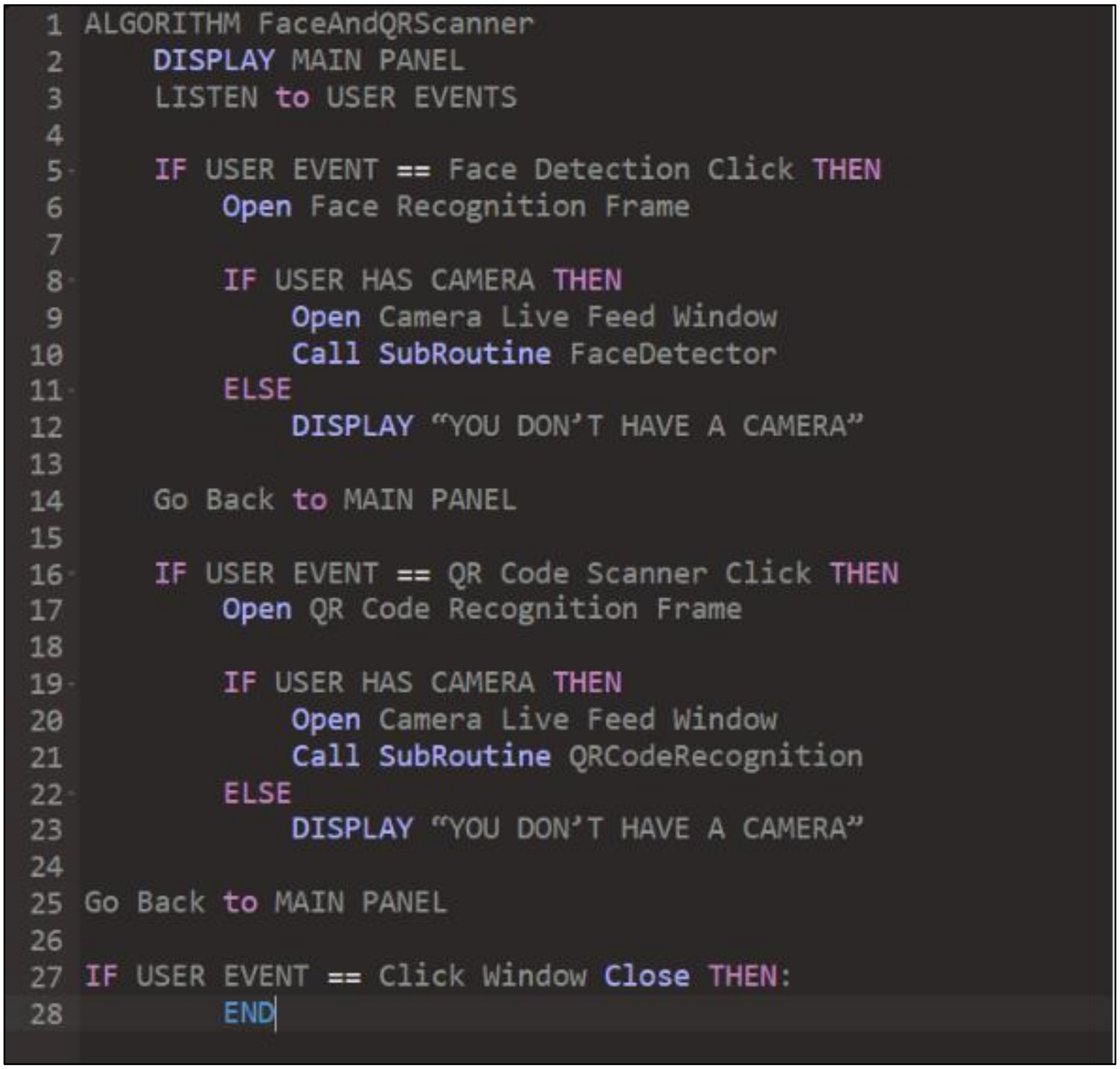
Root Layer of the App

This algorithm shows how the root layer of the application will work. First, it initializes the GUI and display it on the computer screen. Then, it will wait to the user click events.

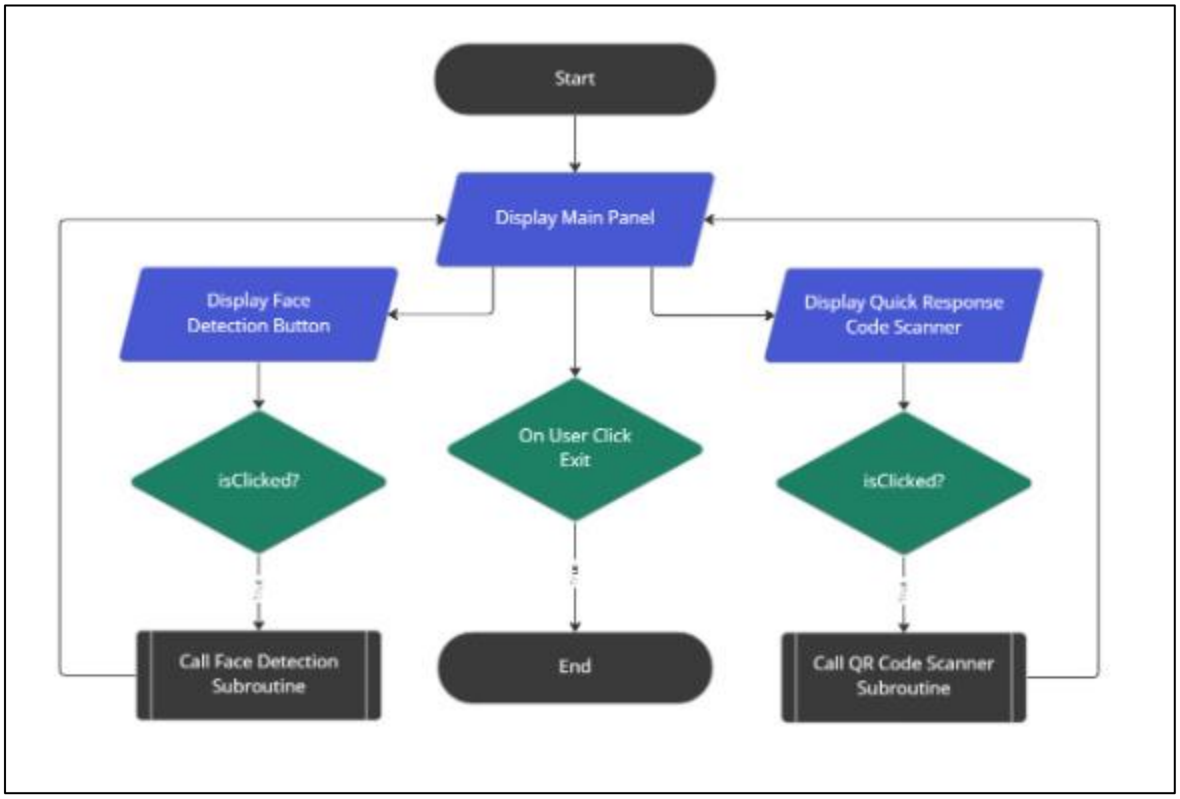
If user clicked the face detection button, the face detection frame will pop up. After that, the system will check if the user has a camera, if so, it will call the Face Detector subroutine to start the face detection. Otherwise, it will display “you don’t have a camera”.

However, if user clicked the QR Code Scanner button instead, the QR code scanner frame will appear. The system will check if user has available camera, the system will call the QR Code Scanner subroutine. If the user does not have available camera, the message box with “you don’t have a camera” text will popup.

The application will exit if user click window ‘X’ button.



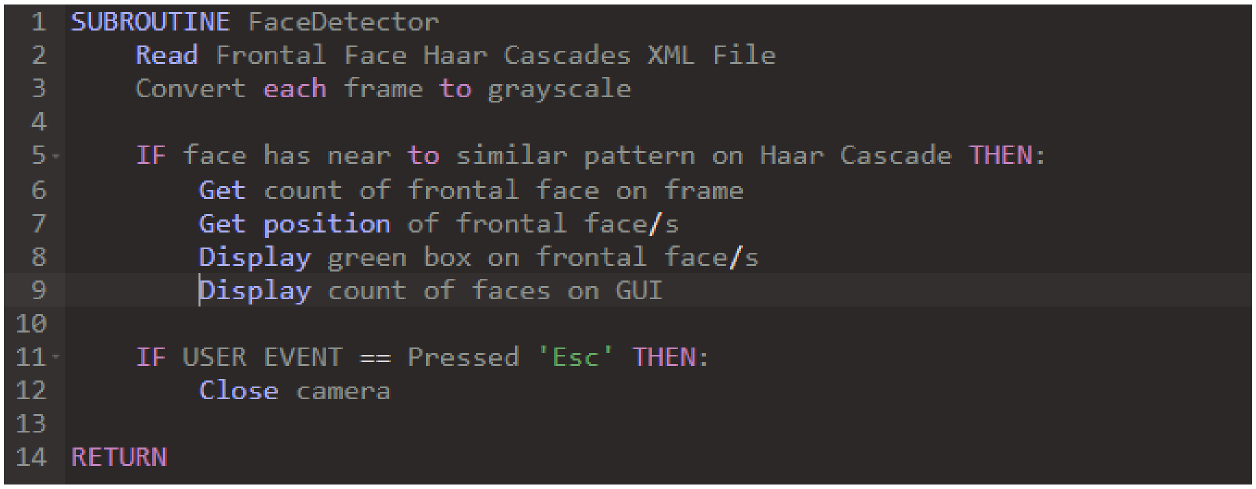
```
ALGORITHM FaceAndQRScanner
    DISPLAY MAIN PANEL
    LISTEN to USER EVENTS

    IF USER EVENT == Face Detection Click THEN
        Open Face Recognition Frame

        IF USER HAS CAMERA THEN
            Open Camera Live Feed Window
            Call SubRoutine FaceDetector
        ELSE
            DISPLAY “YOU DON’T HAVE A CAMERA”

    Go Back to MAIN PANEL

    IF USER EVENT == QR Code Scanner Click THEN
        Open QR Code Recognition Frame

        IF USER HAS CAMERA THEN
            Open Camera Live Feed Window
            Call SubRoutine QRCodeRecognition
        ELSE
            DISPLAY “YOU DON’T HAVE A CAMERA”

Go Back to MAIN PANEL

IF USER EVENT == Click Window Close THEN:
        END
```


Start
Display Main Panel
Display Face Detection Button
Display Quick Response Code Scanner
isClicked?
On User Click Exit
isClicked?
Call Face Detection Subroutine
End
Call QR Code Scanner Subroutine

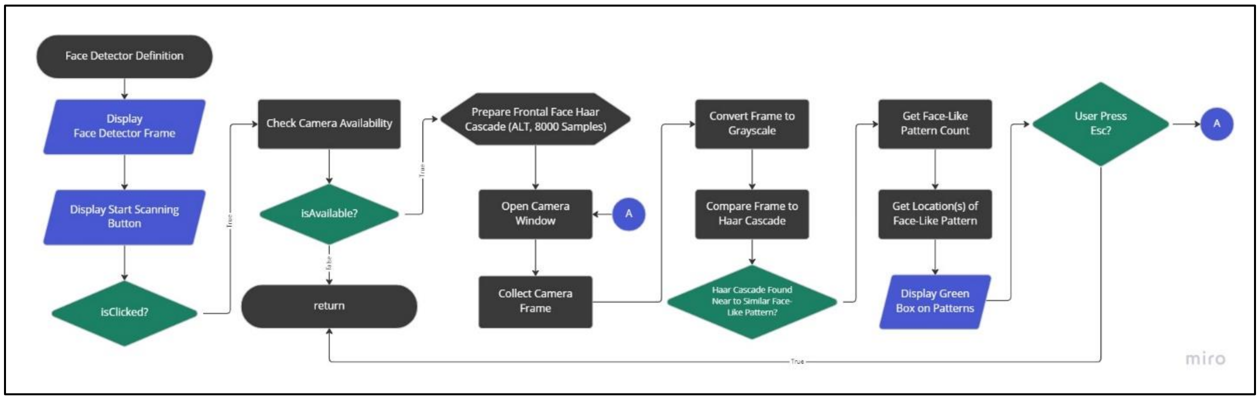
Face Detection Data Frame

This algorithm indicates the process flow of the face detection data frame. Initially, it reads the frontal face Haar Cascade's XML file. Then, each frame being grabbed from the camera stream will be converted into grayscale for better accuracy. Then, if the face has near to similar pattern on Haar Cascade. After that, the logic will count and get the position of the detected frontal face on the frame. Then, the green box will also appear based on the position of faces. Lastly, it will display the count of faces on the GUI.

If user decided to click the 'Esc' button, the camera and current frame will exit. Then the user will be navigated to the main panel of the system again.



```
SUBROUTINE FaceDetector
    Read Frontal Face Haar Cascades XML File
    Convert each frame to grayscale

    IF face has near to similar pattern on Haar Cascade THEN:
        Get count of frontal face on frame
        Get position of frontal face/s
        Display green box on frontal face/s
        Display count of faces on GUI

    IF USER EVENT == Pressed 'Esc' THEN:
        Close camera

RETURN
```

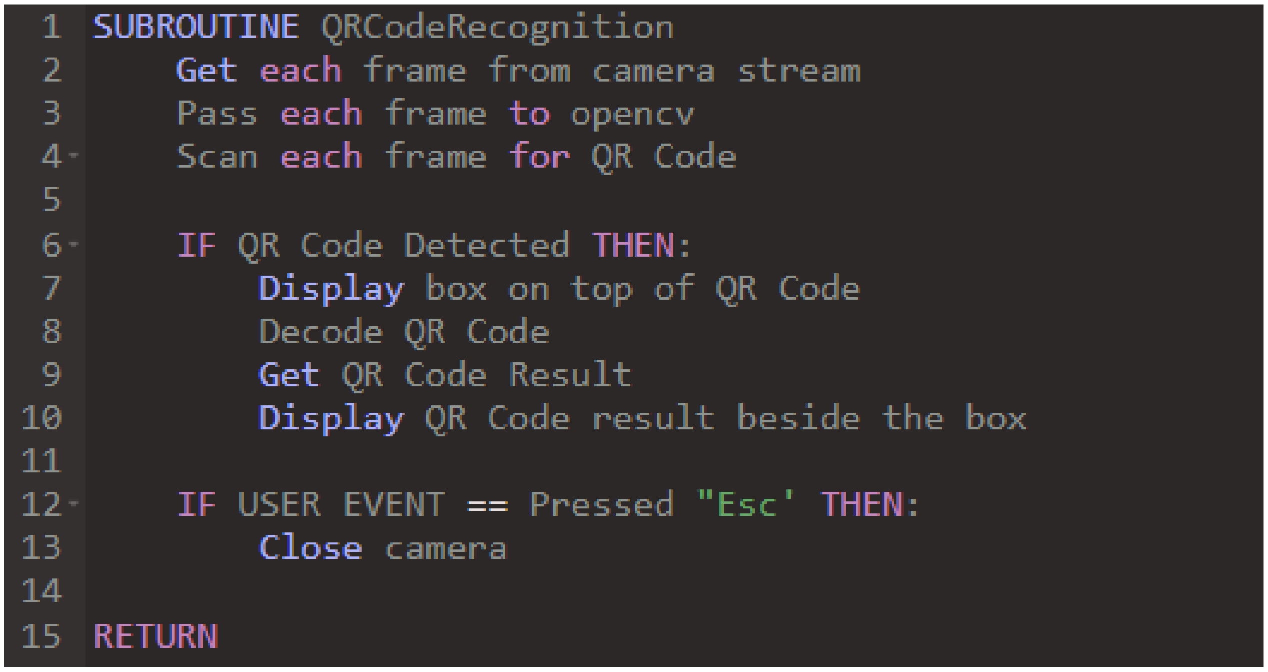


Face Detector Definition
Display Face Detector Frame
Display Start Scanning Button
isClicked?
True
Check Camera Availability
isAvailable?
True
false
return
Prepare Frontal Face Haar Cascade (ALT, 8000 Samples)
Open Camera Window
A
Collect Camera Frame
Convert Frame to Grayscale
Compare Frame to Haar Cascade
Haar Cascade Found Near to Similar Face-Like Pattern?
Get Face-Like Pattern Count
Get Location(s) of Face-Like Pattern
Display Green Box on Patterns
User Press Esc?
A
True
miro

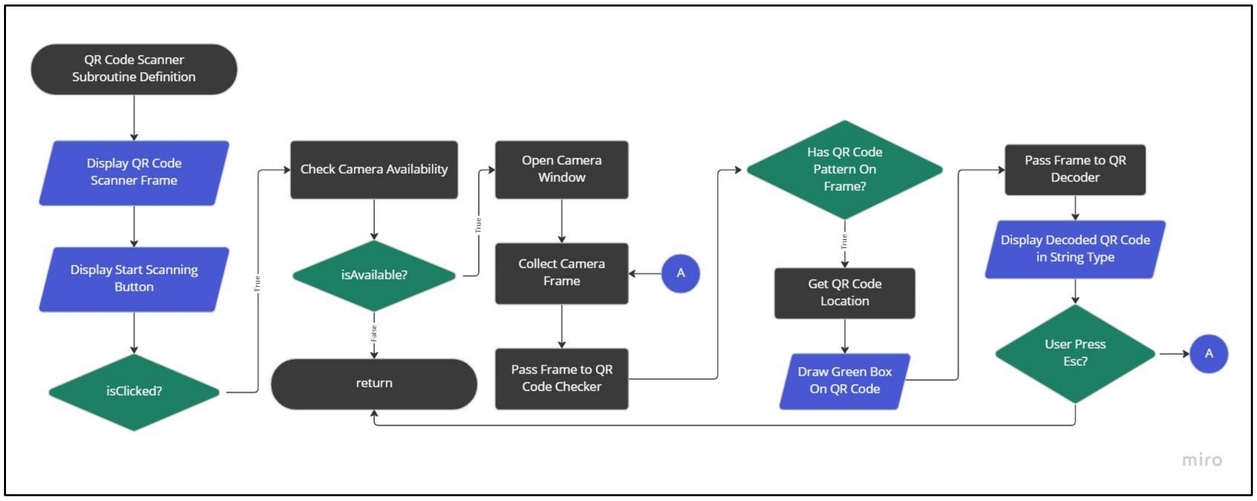
QR Code Scanner Data Frame

This algorithm shows the flow of the Quick Response scanner data frame. The logic behind this feature is basically grabs each frame from the camera stream and it is being passed to the open-source computer vision (open-cv) library and scan the frame if there are any QR code. If there is QR code, then the system will get the position of the QR code and display a box on top of it. Moreover, the open-cv will decode the QR code and get the string type result. Lastly, the system will display the result right beside the box.

If user pressed ‘Esc’, then it will close the camera and will go back to the main frame.



```
SUBROUTINE QRCodeRecognition
    Get each frame from camera stream
    Pass each frame to opencv
    Scan each frame for QR Code

    IF QR Code Detected THEN:
        Display box on top of QR Code
        Decode QR Code
        Get QR Code Result
        Display QR Code result beside the box

    IF USER EVENT == Pressed "Esc' THEN:
        Close camera

RETURN
```


QR Code Scanner Subroutine Definition
Display QR Code Scanner Frame
Display Start Scanning Button
isClicked?
True
Check Camera Availability
isAvailable?
True
False
return
Open Camera Window
Collect Camera Frame
A
Pass Frame to QR Code Checker
Has QR Code Pattern On Frame?
True
Get QR Code Location
Draw Green Box On QR Code
Pass Frame to QR Decoder
Display Decoded QR Code in String Type
User Press Esc?
A
miro

Libraries

1. OpenCV - a free and open-source computer vision library that provides tools for image and video processing.

1.1 Frontal Face Haarcascade V2 – a model of nine thousand face samples compiled in one XML file for object detection and face recognition functions.

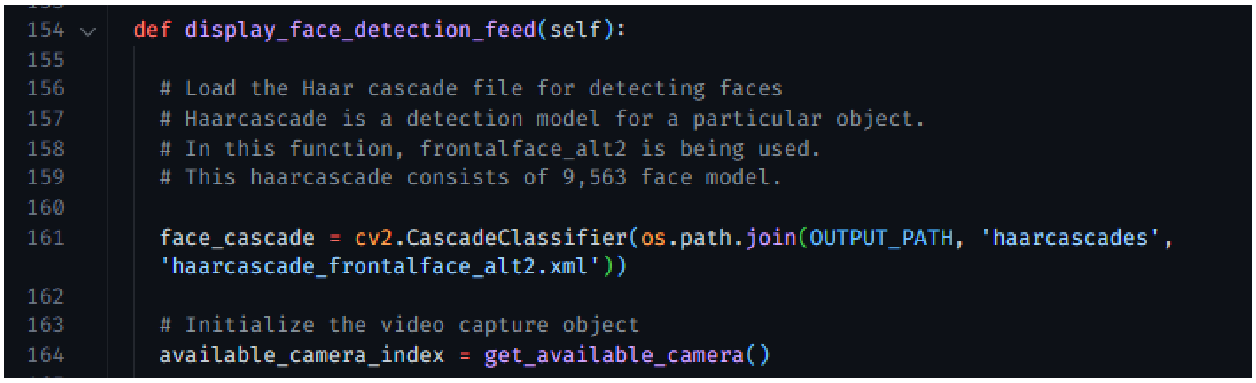
2. TKinter - is a Python library for building graphical user interfaces (GUIs) with the Tk toolkit. It provides tools for creating buttons, labels, text boxes, and menus, as well as support for layout management and event handling.

Project Features Source Code (Back-end)

This portion discusses the back-end source code of the Face Detection and QR Code Scanner feature. The developers ensures that the source code is properly commented and the tried to manage to make it understandable to anyone.

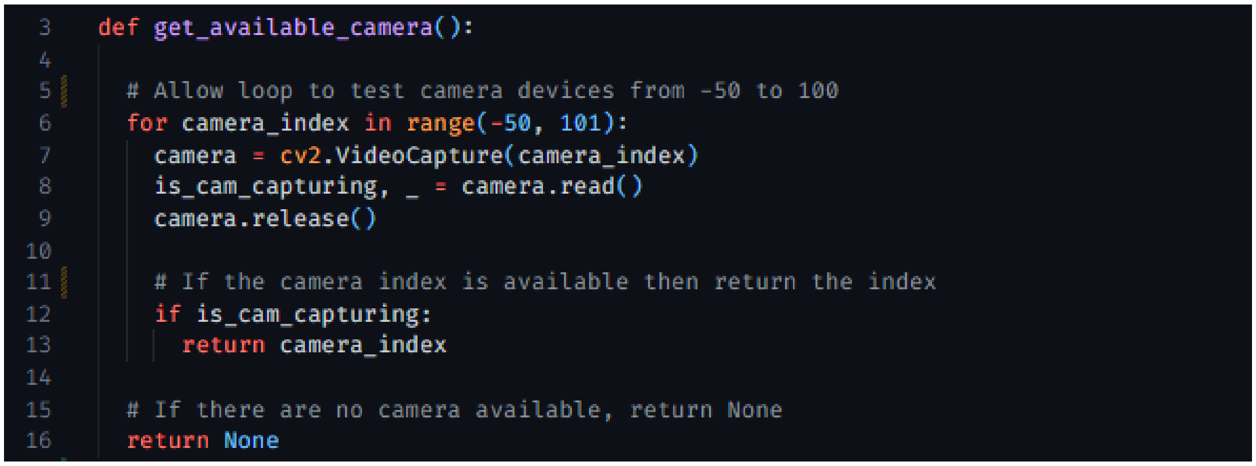
Face Detection Feature

Initialization of frontal face haarcascade second alternative



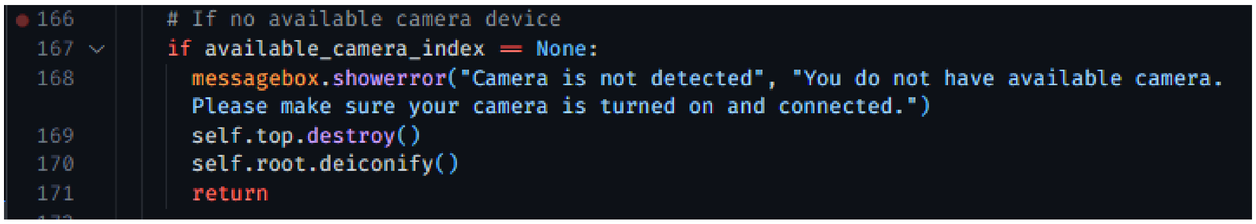
```python
def display_face_detection_feed(self):

    # Load the Haar cascade file for detecting faces
    # Haarcascade is a detection model for a particular object.
    # In this function, frontalface_alt2 is being used.
    # This haarcascade consists of 9,563 face model.

    face_cascade = cv2.CascadeClassifier(os.path.join(OUTPUT_PATH, 'haarcascades',
    'haarcascade_frontalface_alt2.xml'))

    # Initialize the video capture object
    available_camera_index = get_available_camera()
```

Definition of get_available_camera()



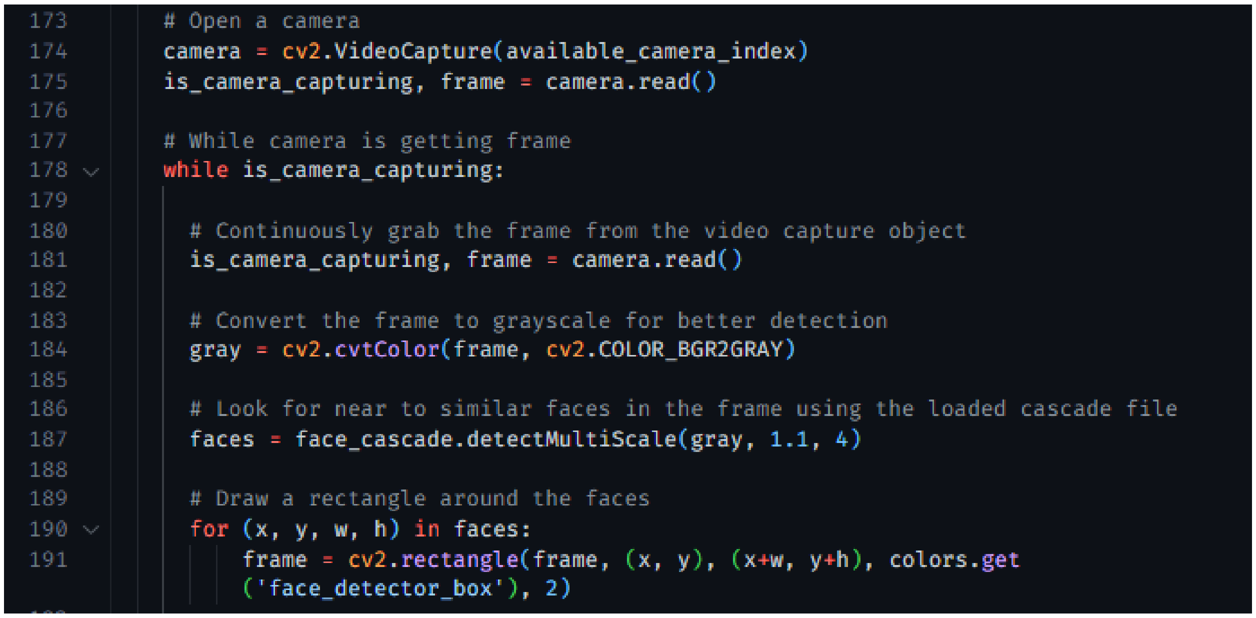
```python
def get_available_camera():

  # Allow loop to test camera devices from -50 to 100
  for camera_index in range(-50, 101):
    camera = cv2.VideoCapture(camera_index)
    is_cam_capturing, _ = camera.read()
    camera.release()

    # If the camera index is available then return the index
    if is_cam_capturing:
      return camera_index

  # If there are no camera available, return None
  return None
```

On get_available_camera() return None



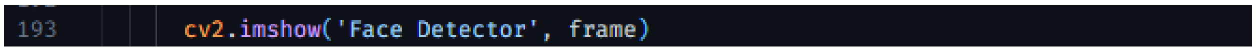
```python
# If no available camera device
if available_camera_index == None:
    messagebox.showerror("Camera is not detected", "You do not have available camera. Please make sure your camera is turned on and connected.")
    self.top.destroy()
    self.root.deiconify()
    return
```

Face Detection under the hood process



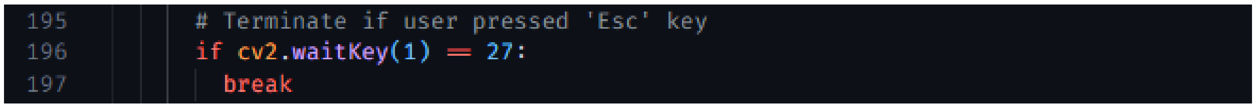
```python
# Open a camera
camera = cv2.VideoCapture(available_camera_index)
is_camera_capturing, frame = camera.read()

# While camera is getting frame
while is_camera_capturing:

    # Continuously grab the frame from the video capture object
    is_camera_capturing, frame = camera.read()

    # Convert the frame to grayscale for better detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Look for near to similar faces in the frame using the loaded cascade file
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        frame = cv2.rectangle(frame, (x, y), (x+w, y+h), colors.get('face_detector_box'), 2)
```

Displaying frames on the camera stream

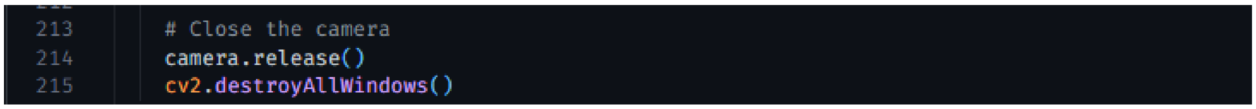


```python
    cv2.imshow('Face Detector', frame)
```

On user termination

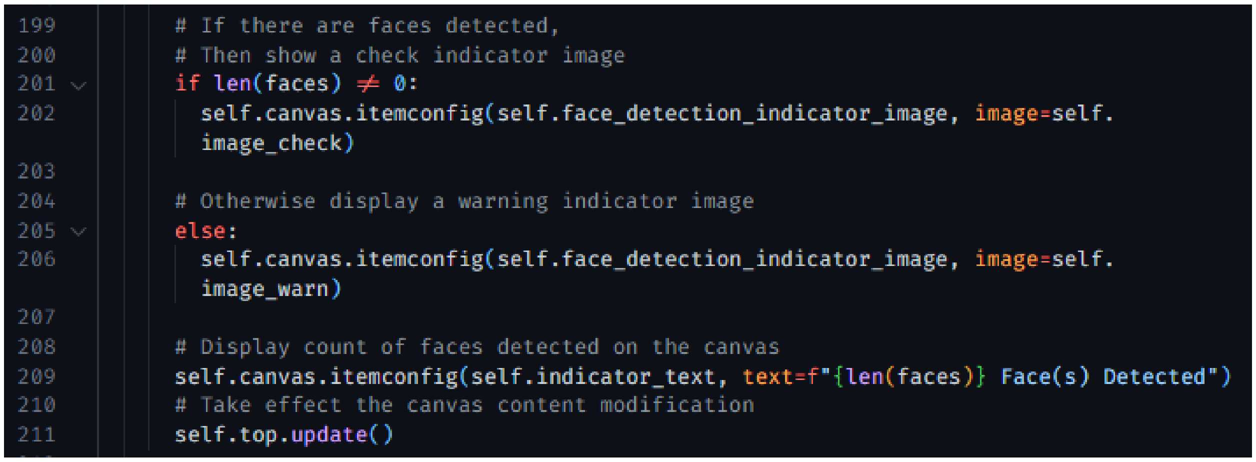


```python
    # Terminate if user pressed 'Esc' key
    if cv2.waitKey(1) == 27:
        break
```



```python
# Close the camera
camera.release()
cv2.destroyAllWindows()
```

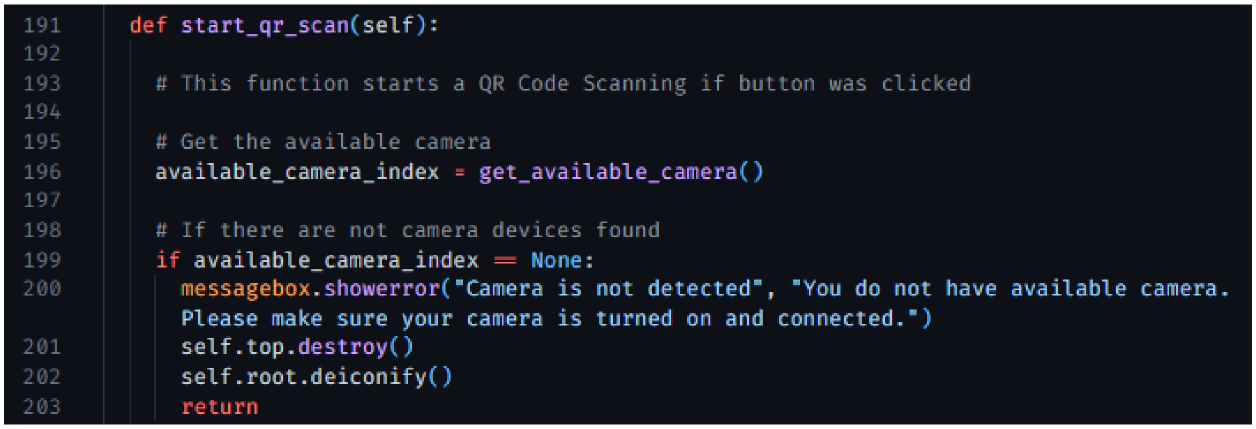
Applying UI modification on the frame



```python
    # If there are faces detected,
    # Then show a check indicator image
    if len(faces) != 0:
        self.canvas.itemconfig(self.face_detection_indicator_image, image=self.image_check)

    # Otherwise display a warning indicator image
    else:
        self.canvas.itemconfig(self.face_detection_indicator_image, image=self.image_warn)

    # Display count of faces detected on the canvas
    self.canvas.itemconfig(self.indicator_text, text=f"{len(faces)} Face(s) Detected")
    # Take effect the canvas content modification
    self.top.update()
```

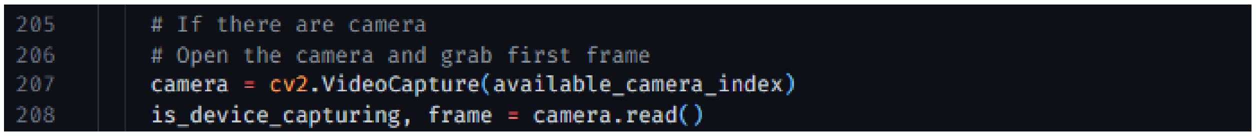
QR Code Scanner Feature

Open a camera stream if user has available camera device



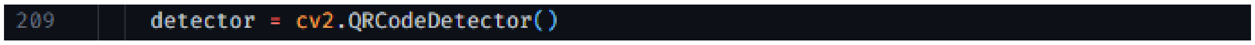
```python
def start_qr_scan(self):

    # This function starts a QR Code Scanning if button was clicked

    # Get the available camera
    available_camera_index = get_available_camera()

    # If there are not camera devices found
    if available_camera_index == None:
        messagebox.showerror("Camera is not detected", "You do not have available camera. Please make sure your camera is turned on and connected.")
        self.top.destroy()
        self.root.deiconify()
        return
```

Grab each frame from the camera



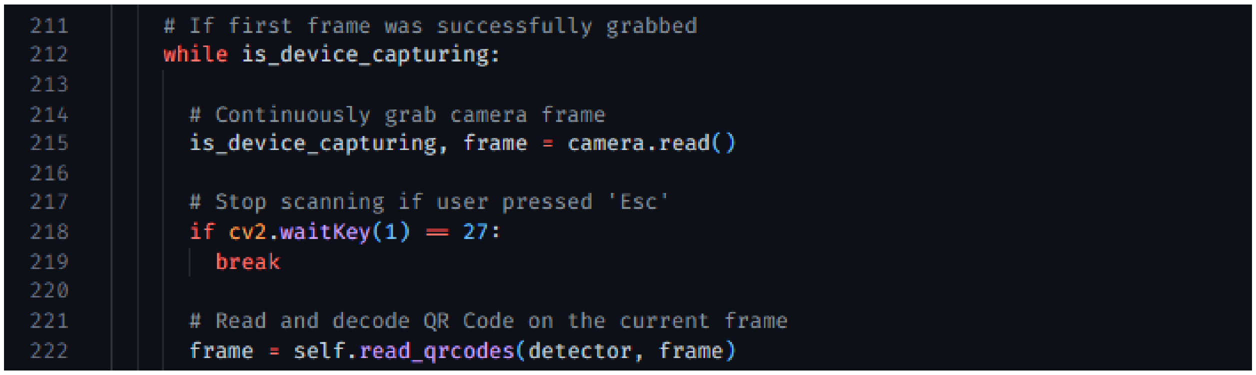
```python
    # If there are camera
    # Open the camera and grab first frame
    camera = cv2.VideoCapture(available_camera_index)
    is_device_capturing, frame = camera.read()
```

Initialize the OpenCV QR Code Detector



```python
    detector = cv2.QRCodeDetector()
```

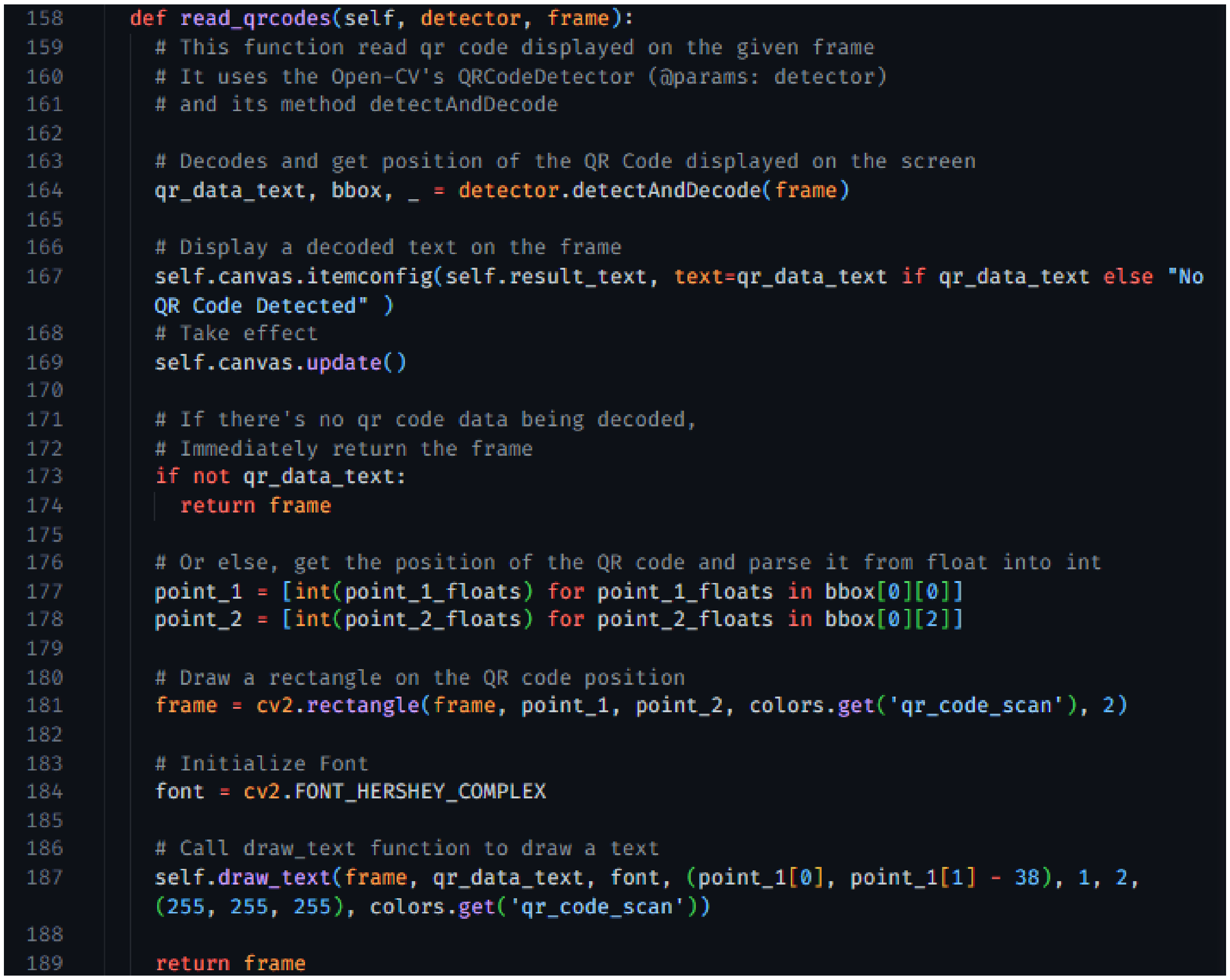
On camera continuous capture



```
# If first frame was successfully grabbed
while is_device_capturing:

    # Continuously grab camera frame
    is_device_capturing, frame = camera.read()

    # Stop scanning if user pressed 'Esc'
    if cv2.waitKey(1) == 27:
        break

    # Read and decode QR Code on the current frame
    frame = self.read_qrcodes(detector, frame)
```

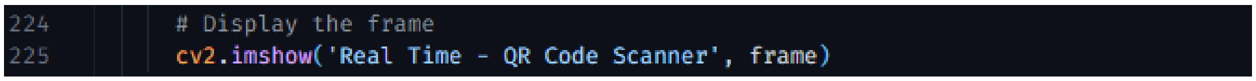
Definition of user-defined read_qrcodes function

* It decodes frame and get the data text and box position
* Send decoded text to GUI frame and update the GUI
* If the frame detects no QR code, return the plain frame
* Else, get the position and draw a rectangle and display text on the screen



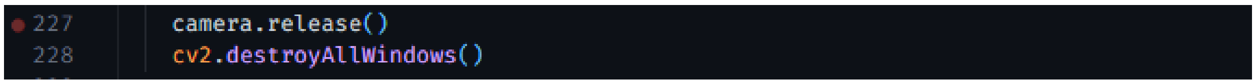
```
def read_qrcodes(self, detector, frame):
    # This function read qr code displayed on the given frame
    # It uses the Open-CV's QRCodeDetector (@params: detector)
    # and its method detectAndDecode

    # Decodes and get position of the QR Code displayed on the screen
    qr_data_text, bbox, _ = detector.detectAndDecode(frame)

    # Display a decoded text on the frame
    self.canvas.itemconfig(self.result_text, text=qr_data_text if qr_data_text else "No QR Code Detected" )
    # Take effect
    self.canvas.update()

    # If there's no qr code data being decoded,
    # Immediately return the frame
    if not qr_data_text:
        return frame

    # Or else, get the position of the QR code and parse it from float into int
    point_1 = [int(point_1_floats) for point_1_floats in bbox[0][0]]
    point_2 = [int(point_2_floats) for point_2_floats in bbox[0][2]]

    # Draw a rectangle on the QR code position
    frame = cv2.rectangle(frame, point_1, point_2, colors.get('qr_code_scan'), 2)

    # Initialize Font
    font = cv2.FONT_HERSHEY_COMPLEX

    # Call draw_text function to draw a text
    self.draw_text(frame, qr_data_text, font, (point_1[0], point_1[1] - 38), 1, 2, (255, 255, 255), colors.get('qr_code_scan'))

    return frame
```

Displaying the returned frame from read_qrcodes function on the video stream



```python
    # Display the frame
    cv2.imshow('Real Time - QR Code Scanner', frame)
```

On user pressed ‘Esc’ (refer to line 218), stop capturing and destroy windows



```python
    camera.release()
    cv2.destroyAllWindows()
```

**** END OF THE DOCUMENTATION ****

Addendum

Decode Team as Developer

The source-code of this project was developed by the Group Four (4) of the sophomore students’ section three (3) of the Bachelor of Science in Computer Science in Polytechnic University of the Philippines. This project is designed, coded, and developed duly for the academic purposes only. The team “Decode” managed to choose open-source assets and libraries in accordance with the best of our knowledge. Any critiques and suggestions are open for the public.

Source Code released to the Public

The team “Decode” allows anyone to clone and download the source-code of this project under certain conditions:

1. Anyone can modify, edit, and improve the content of the system.
2. We, “Decode” team, are not accountable to any damages, if this system can harm, to any hardware or software.
3. THE SOFTWARE IS PROVIDED "AS IS".

Source Code Link: https://github.com/markcalendario/face-detector-and-qrcode-scanner